Importing the Dependencies

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

In [2]:
```
# Loading the csv data to a Pandas DataFrame
diabetes_data= pd.read_excel('F:\Final year project\INTERNSHIP\diabetes data.xlsx')
# Read Raw Dataset
diabetes_data.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes	PedigreeFunction	Age
0	6	6	148	72	35	0	33.6	0.627	50
1	0	1	85	66	29	0	26.6	0.351	31
2	9	8	183	64	0	0	23.3	0.672	32
3	7	1	89	66	23	94	28.1	0.167	21
4	3	0	137	40	35	168	43.1	2.288	33

In [4]:
```
# print last 5 rows of the dataset
diabetes_data.tail()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes	PedigreeFunction	A
763	4	10	101	76	48	180	32.9	0.171	
764	1	2	122	70	27	0	36.8	0.340	
765	4	5	121	72	23	112	26.2	0.245	
766	4	1	126	60	0	0	30.1	0.349	
767	8	1	93	70	31	0	30.4	0.315	

In [5]:
```
# number of rows and columns in the dataset
diabetes_data.shape
```

Out[5]:
```
(768, 10)
```

In [6]:
```
# getting some info about the data
diabetes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Pregnancies           768 non-null    int64  
 1   Glucose               768 non-null    int64  
 2   BloodPressure         768 non-null    int64  
 3   SkinThickness         768 non-null    int64  
 4   Insulin               768 non-null    int64  
 5   BMI                   768 non-null    int64  
 6   Diabetes              768 non-null    float64
 7   PedigreeFunction      768 non-null    float64
 8   Age                   768 non-null    int64  
 9   Outcome               768 non-null    int64  
dtypes: float64(2), int64(8)
memory usage: 60.1 KB
```

In [7]:
```
# checking for missing values
diabetes_data.isnull().sum()
```

Out[7]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
Diabetes                    0
PedigreeFunction            0
Age                         0
Outcome                     0
dtype: int64
```

In [8]:
```
# statistical measures about the data
diabetes_data.describe()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes	Pedig
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	5.065104	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.307127	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	5.000000	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	8.000000	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	10.000000	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [10]:
```
# checking the distribution of Target Variable

diabetes_data['Outcome'].value_counts()
```

Out[10]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [12]:
```
X = diabetes_data.drop(columns='Outcome', axis=1)
Y = diabetes_data['Outcome']
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  \
0              6        6            148             72       35    0
1              0        1             85             66       29    0
2              9        8            183             64        0    0
3              7        1             89             66       23   94
4              3        0            137             40       35  168
..           ...      ...            ...            ...      ...  ...
763            4       10            101             76       48  180
764            1        2            122             70       27    0
765            4        5            121             72       23  112
766            4        1            126             60        0    0
767            8        1             93             70       31    0

     Diabetes  PedigreeFunction  Age
0        33.6             0.627   50
1        26.6             0.351   31
2        23.3             0.672   32
3        28.1             0.167   21
4        43.1             2.288   33
..        ...               ...  ...
763      32.9             0.171   63
764      36.8             0.340   27
765      26.2             0.245   30
766      30.1             0.349   47
767      30.4             0.315   23

[768 rows x 9 columns]
```

In [13]:
```
print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [14]:
```
diabetes_data['Outcome'].value_counts()
```

Out[14]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [15]:
```
diabetes_data.groupby('Outcome').mean()
```

Out[15]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes	Pedigr
0	4.898000	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	
1	5.376866	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	

In [16]:
```
X = diabetes_data.drop(columns = 'Outcome', axis=1)
Y =diabetes_data['Outcome']
```

In [17]:
```
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  \
0              6        6            148             72       35    0
1              0        1             85             66       29    0
2              9        8            183             64        0    0
3              7        1             89             66       23   94
4              3        0            137             40       35  168
..           ...      ...            ...            ...      ...  ...
763            4       10            101             76       48  180
764            1        2            122             70       27    0
765            4        5            121             72       23  112
766            4        1            126             60        0    0
767            8        1             93             70       31    0

     Diabetes  PedigreeFunction  Age
0        33.6             0.627   50
1        26.6             0.351   31
2        23.3             0.672   32
3        28.1             0.167   21
4        43.1             2.288   33
..        ...               ...  ...
763      32.9             0.171   63
764      36.8             0.340   27
765      26.2             0.245   30
766      30.1             0.349   47
767      30.4             0.315   23

[768 rows x 9 columns]
```

In [18]:
```
print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [19]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, ra
print(X.shape, X_train.shape, X_test.shape)
```

```
(768, 9) (614, 9) (154, 9)
```

In [20]:
```
classifier = svm.SVC(kernel='linear')
```

In [21]:
```
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

Out[21]:
```
SVC(kernel='linear')
```

In [22]:
```
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [23]:
```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  0.7899022801302932
```

In [24]:
```
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7662337662337663
```

In [25]:
```
input_data = (10,1,85,66,29,0,26.6,0.351,31)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = classifier.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The person doesnt have diabetes')
else:
  print('The person has diabetes')
```

```
[0]
The person doesnt have diabetes
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```